2020 年新工科联盟-Xilinx 暑期学校团队项目设计文档

设计文稿提交格式

(Project Paper Submission Template)

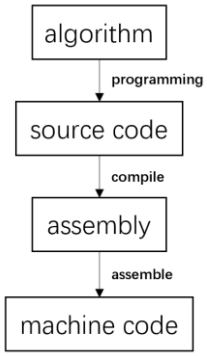
设计作品名称	基于 SEA Board 的 RISC-V 架构处理器软核移植
参赛队员姓名、学号、学校及所在院系	招梓枫，22017327，仪器科学与工程学院，东南大学 王睿彪，22017110，仪器科学与工程学院，东南大学
Github 链接	https://github.com/zzFon/RISC_on_FPGA

第一部分

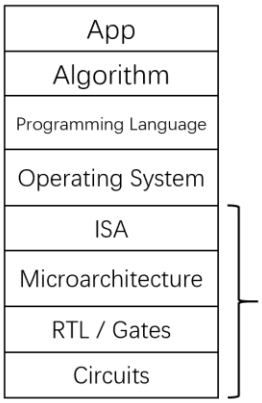
设计概述 /Design Introduction

（请概括地描述一下你的设计，包括设计目的、应用领域及适用范围等。撰写过程中应注重突出设计实现的主要/特色功能）

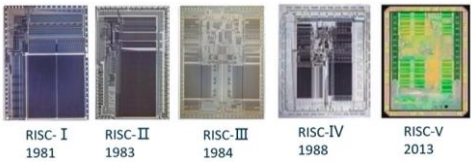
芯片(chip)是整个电子信息产业的基石，而微处理器（MPU, Micro Processor Unit）则是芯片产业中举足轻重的一部分.微处理器作为一种特殊的芯片，承担着电子系统中主要的信息处理与控制任务.微处理器涉及的技术众多，包括半导体、超大规模集成电路(VLSI, Very Large Scale Integrated circuit)、指令集架构(ISA, Instruction Set Architecture)、计算机组成等多个不同层次的研究领域.



从算法到机器码



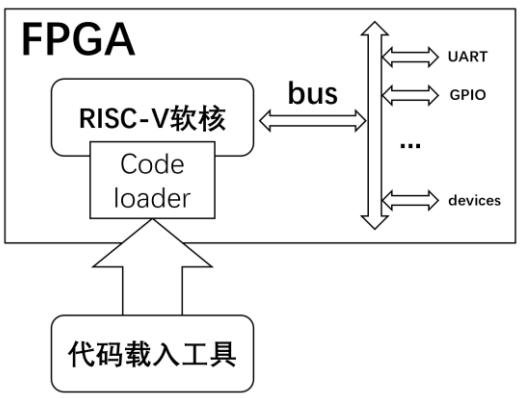
计算机体系结构



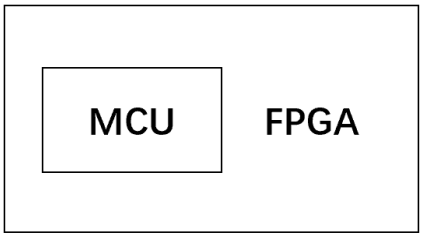
RISC-V 架构

其中，指令集架构 ISA 作为对微处理器操作指令的定义，连接了操作系统(OS, Operating System)和微架构(Microarchitecture)，在计算机体系结构中起到承上启下的作用.目前常用的指令集架构分主要包括：基于 Intel 8086 家族的 x86 架构、移动设备微处理器大量使用的 ARM 架构、风靡一时的 MIPS 架构，以及完全开源共享的 RISC-V 架构等.

RISC-V 架构是由加州大学伯克利分校科研人员再 2010 年发明的一套新的 ISA 架构,相比无法获得的 x86 架构 ISA 源码和费用高昂的 ARM 架构专利授权,RISC-V 完全开源共享.在设计上,RISC-V 不像 x86 和 ARM 架构那样需要考虑兼容问题而导致复杂和冗余的设计,在部署上,RISC-V 也已经有了一套完整架构文档和编译器等 CPU 软件工具链.RISC-V 的前景相当光明,而对于我国企业近年来屡次受到美国的芯片技术封锁,RISC-V 在中国更是大有用武之地.



RISC-V 软核移植到 FPGA



在 FPGA 中通过软核移植内嵌 MCU

本次项目设计的内容为基于 SEA Board 进行 RISC-V 架构处理器软核的移植.通过处理器软核移植,在 SEA Board 的 Xilinx Spartan-7 系列 FPGA 上实现一个 PIC 系列 MCU,并在其之上运行 C 语言程序,从而进行验证.

项目设计特点:

- 了解 RISC 以及 RISC-V 架构
- 在 FPGA 上实现微处理器软核
- 了解指令集架构的软核移植方法
- 处理器规模小,在 Xilinx Spartan-7 系列 FPGA 上 LUT 资源占用率低

第二部分

系统组成及功能说明 /System Construction & Function Description

（请详细说明你作品要实现的所有功能以及如何组建系统以实现该功能，还包括为实现该功能需要用到的所有参数和所有操作的详细说明，必要的地方多用图表形式表述）

1. PIC 系列 MCU

软核移植后,要求实现 Microchip 公司系列的 MCU:PIC16C56,并在其上运行 C 语言程序进行测试.PIC16C56 是一款高性能 RISC CPU,关于 PIC16C56 的部分信息如下[14]:

- 指令集只有 33 条单字指令
- 多数指令在单周期完成
- 12-bit 宽的指令
- 8-bit 宽字长
- 对于数据和指令均支持直接寻址,间接寻址和相对寻址模型
- 低功耗,并采用高速 CMOS EPROM/ROM 工艺

TABLE 1-1: PIC16C5X FAMILY OF DEVICES

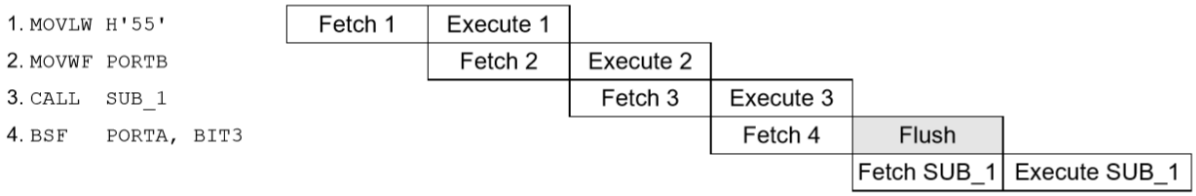
Features	PIC16C54	PIC16CR54	PIC16C55	PIC16C56	PIC16CR56
Maximum Operation Frequency	40 MHz	20 MHz	40 MHz	40 MHz	20 MHz
EPROM Program Memory (x12 words)	512	—	512	1K	—
ROM Program Memory (x12 words)	—	512	—	—	1K
RAM Data Memory (bytes)	25	25	24	25	25
Timer Module(s)	TMR0	TMR0	TMR0	TMR0	TMR0
I/O Pins	12	12	20	12	12
Number of Instructions	33	33	33	33	33
Packages	18-pin DIP, SOIC; 20-pin SSOP	18-pin DIP, SOIC; 20-pin SSOP	28-pin DIP, SOIC; 28-pin SSOP	18-pin DIP, SOIC; 20-pin SSOP	18-pin DIP, SOIC; 20-pin SSOP
All PIC® Family devices have Power-on Reset, selectable Watchdog Timer, selectable Code Protect and high I/O current capability.					

PIC16C5X 系列 MCU 参数表

Features	PIC16C57	PIC16CR57	PIC16C58	PIC16CR58
Maximum Operation Frequency	40 MHz	20 MHz	40 MHz	20 MHz
EPROM Program Memory (x12 words)	2K	—	2K	—
ROM Program Memory (x12 words)	—	2K	—	2K
RAM Data Memory (bytes)	72	72	73	73
Timer Module(s)	TMR0	TMR0	TMR0	TMR0
I/O Pins	20	20	12	12
Number of Instructions	33	33	33	33
Packages	28-pin DIP, SOIC; 28-pin SSOP	28-pin DIP, SOIC; 28-pin SSOP	18-pin DIP, SOIC; 20-pin SSOP	18-pin DIP, SOIC; 20-pin SSOP
All PIC® Family devices have Power-on Reset, selectable Watchdog Timer, selectable Code Protect and high I/O current capability.				

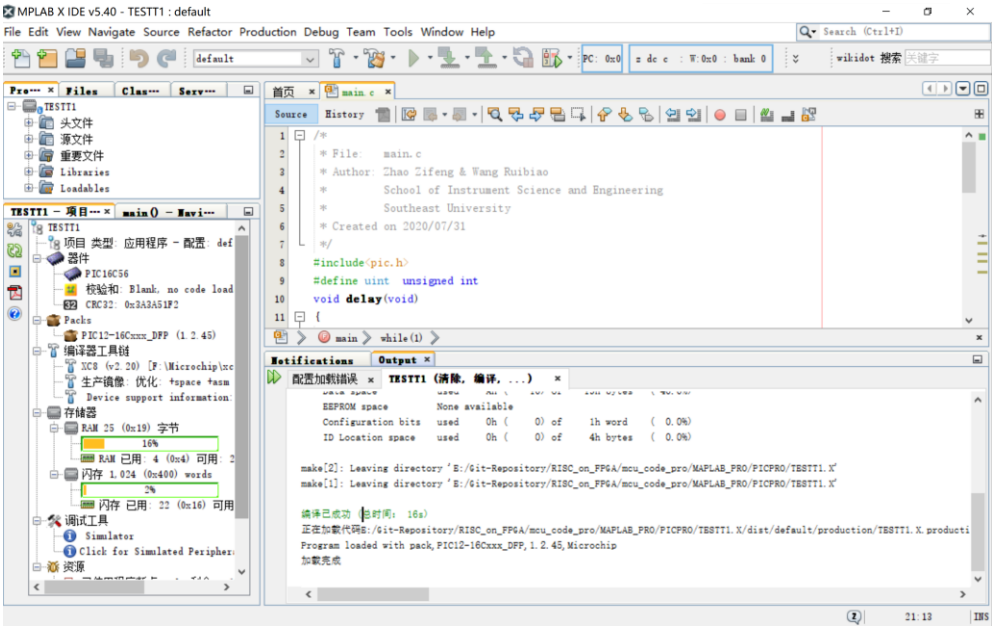
PIC16C5X 系列 MCU 参数表

PIC16C5X 系列 MCU 采用 2 级流水线:取指,执行,使得所有机器指令(program branches 指令除外)均能在 1 个 CPU 周期完成



PIC16C5X 系列 MCU 流水线

使用 Microchip 公司的 PIC 开发工具 MPLAB X IDE 进行开发,并用 XC8 编译器对 C 语言程序进行编译得到.hex 文件

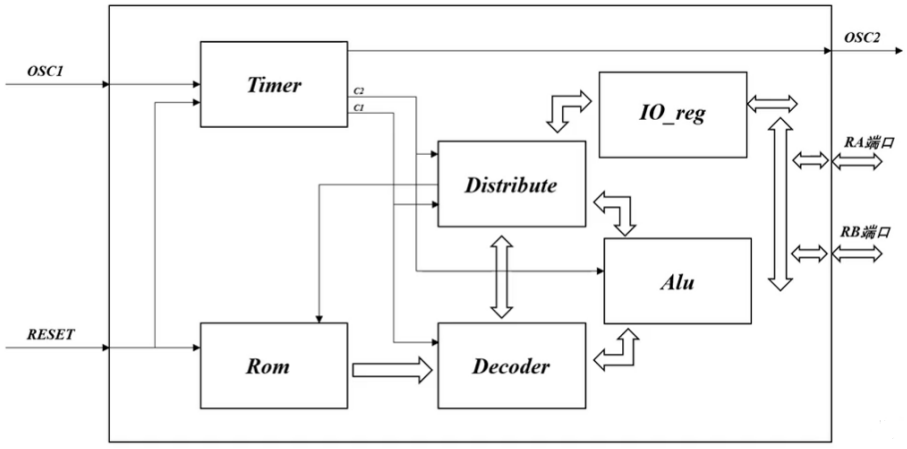


使用 MPLAB X IDE 进行 PIC16C5X 开发

2. 处理器的软核结构

所使用的 RISC-V 软核包含 5 个主要模块:

- timer:产生 MCU 工作的时序信号
- ROM:存储编译器(compiler)编译生成的指令
- decoder:对指令进行译码
- distribute:控制指令执行
- ALU:算术逻辑单元
- IO_reg:负责 IO 端口的控制



所用软核组织结构的原理图[6]

3. 移植工序与工具链

Step1. 工程文件结构介绍

---core: RISC 内核工程

------project_1.xpr: vivado 项目工程文件

------pic16c56.v: 顶层模块

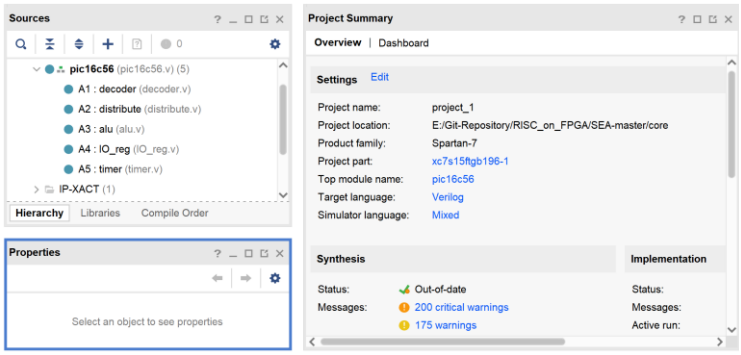
------alu.v: ALU 模块

------decoder.v: decoder 模块

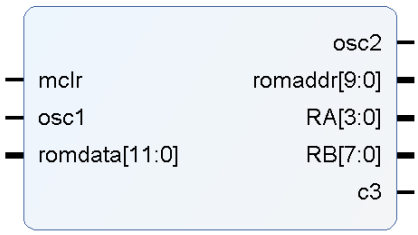
------distribute.v：distribute 模块
------IO_reg.v：IO_reg 模块
---SEA：将内核移植到 SEA Board 所使用的工程
------project_5.xpr：vivado 工程文件
------project_5.srcs
---------sources_1
------------new
---------------top.v：顶层模块
---MPLAB_PRO：MCU 所运行程序的工程
------PICPRO
---------TESTT1.X
------------main.c：MCU 上运行的程序
--------------dist
----------------default
------------------production
--------------------TESTT1.X.production.hex：c 源程序编译得到的.hex 文件

Step2. RISC 内核准备

上文已经介绍了所用的 RISC-V 软核的组织结构。编辑 RISC 内核工程(core)，注意设定 FPGA 型号，Synthesis、Implementation、Simulation 确认无误后，封装成 IP 核：pic16c_56_v1



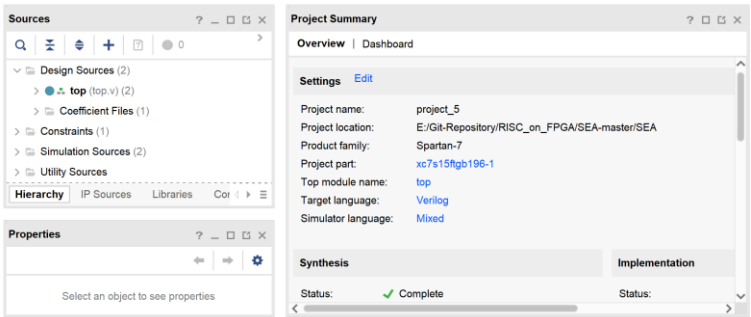
RISC 内核的 core 工程



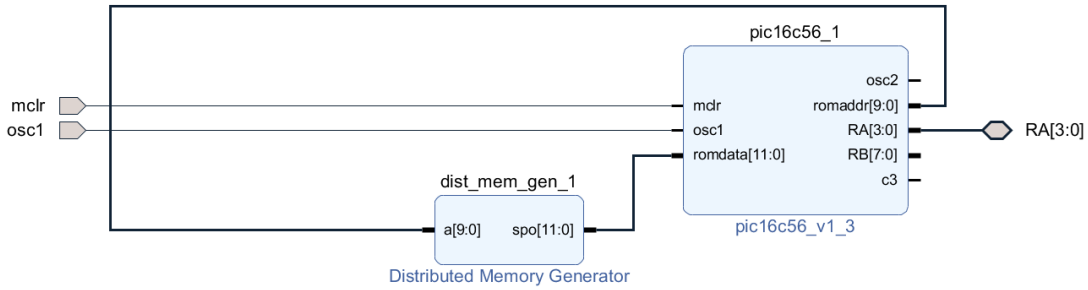
将内核封装成 IP

Step3. RISC 内核移植(IP 核调用)

编辑移植工程(SEA)，同样的，注意设定 FPGA 型号。top.v 顶层模块下，调用 core 工程封装得到的 IP：pic16c_56_v1，另外调用一个 ROM IP：dist_mem_gen_1。如图，连接 2 个模块。



SEA 工程



连接内核、ROM、端口

Step4. MCU 程序设计

建立 MPLAB 工程,并为 MPLAB X IDE 插入 8-bit 的 PIC 系列 MCU 专用编译器 XC8[12][13][14]。编辑 main.c 源程序，如图所示进行 MCU 程序设计，利用 SEA Board 上的 2 盏 LED 灯以 2 进制计数的方式进行亮灭控制。

```
/*
 * File:   main.c
 * Author: Zhao Zifeng & Wang Ruibiao
 *         School of Instrument Science and Engineering
 *         Southeast University
 * Created on 2020/07/31
 */
#include<pic.h>
#define uint  unsigned int
void delay(void)
{
    uint a,b,c;
    for(a=200;a>0;a--)
        for(b=110;b>0;b--)
            for(c=100;c>0;c--);
}

void main()
{
OPTION=8;
TRISA=0;
    while(1)
    {
        PORTA = 1;//0001
        delay();
        PORTA = 2;//0010
        delay();
        PORTA = 3;//0011
        delay();
        PORTA = 0;//0000
        delay();
    }
}
```

在 MPLAB 工程中使用 XC8 编译器进行编译，得到.hex 文件，用于后续操作。

Step5. ROM 的初始化

通过.hex 文件与.coe 文件转换器将上文获得的的.hex 文件转换为.coe 文件，在移植工程(SEA)中引用该.coe 文件，用于初始化 ROM。

Step6. 约束设计、综合

依据 SEA Board 的原理图[15]和外围 IO 接口说明[16]选用合适的管教，并在 SEA 工程中定义.xdc 约束文件，为顶层模块的端口绑定管脚。依次进行 Synthesis、Implementation、Simulation，确认无误后进行 Bitstream Generation 得到 top.bit 文件

Step7. 烧录

万事俱备，只欠东风，最激动人心的时刻来了。按照 SEA Board 实验指导书[17]流程将 top.bit 文件烧录到 FPGA 上,按下板上 RST 按钮进行复位,支持 8-bitPIC 系列 MCU 的 RISC-V 架构软核就被移植到了 FPGA 上，重复 Step4~7 即可在 FPGA 的内嵌 pic16c_56 上运行更复

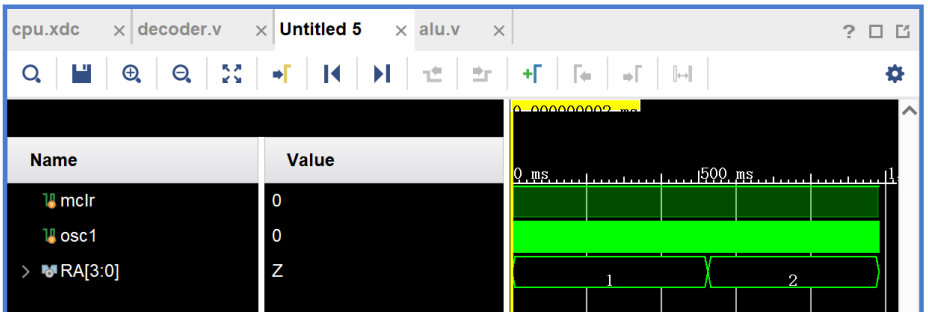
杂的 MCU 程序！

第三部分

完成情况及性能参数 /Final Design & Performance Parameters

完成情况：分模块完成作品（已实现的功能）：

1. 项目仿真图

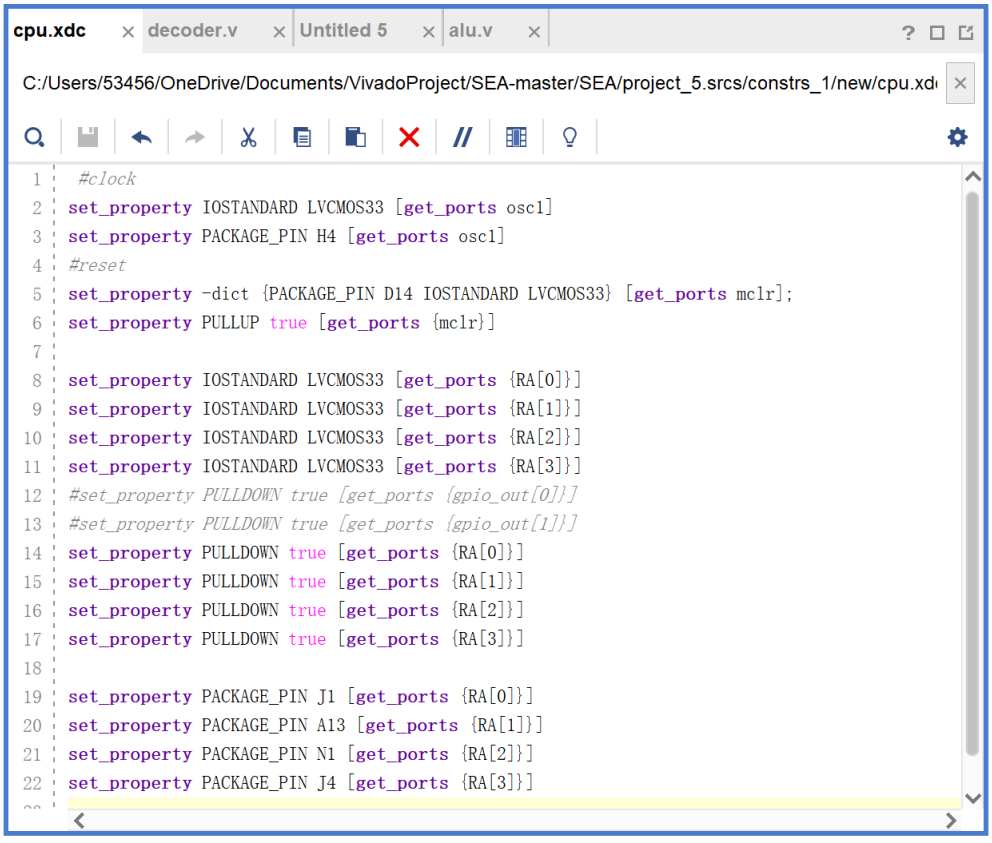


mclr 为使能信号，高电平使能，低电平复位；

osc1 为时钟信号；

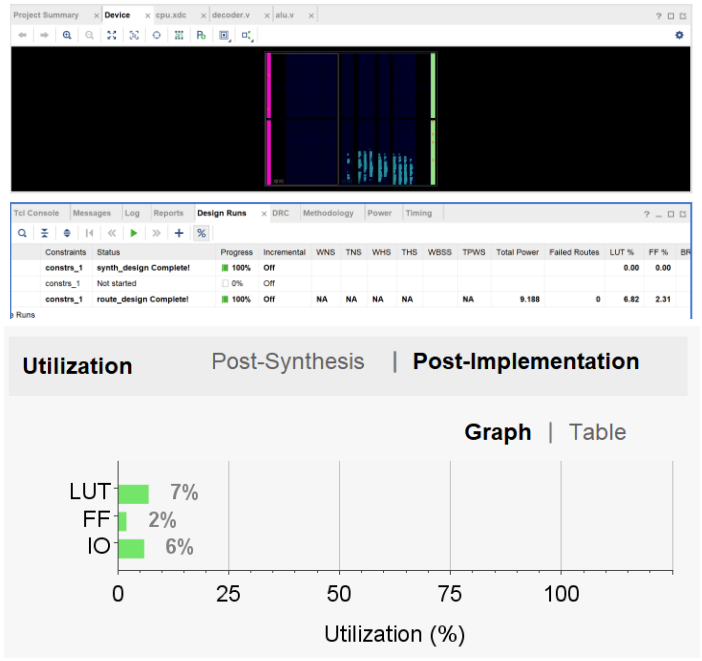
RA 为输出信号，RA0 和 RA1 分别接在两个 LED 上。

2. 引脚分配约束文件(.xdc)

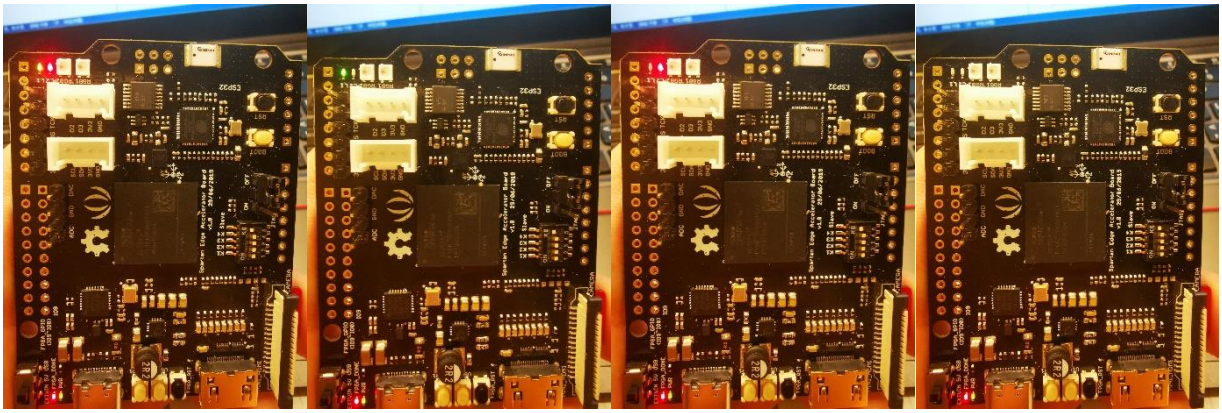



```
1  #clock
2  set_property IOSTANDARD LVCMOS33 [get_ports osc1]
3  set_property PACKAGE_PIN H4 [get_ports osc1]
4  #reset
5  set_property -dict {PACKAGE_PIN D14 IOSTANDARD LVCMOS33} [get_ports mclr];
6  set_property PULLUP true [get_ports {mclr}]
7
8  set_property IOSTANDARD LVCMOS33 [get_ports {RA[0]}]
9  set_property IOSTANDARD LVCMOS33 [get_ports {RA[1]}]
10 set_property IOSTANDARD LVCMOS33 [get_ports {RA[2]}]
11 set_property IOSTANDARD LVCMOS33 [get_ports {RA[3]}]
12 #set_property PULLDOWN true [get_ports {gpio_out[0]}]
13 #set_property PULLDOWN true [get_ports {gpio_out[1]}]
14 set_property PULLDOWN true [get_ports {RA[0]}]
15 set_property PULLDOWN true [get_ports {RA[1]}]
16 set_property PULLDOWN true [get_ports {RA[2]}]
17 set_property PULLDOWN true [get_ports {RA[3]}]
18
19 set_property PACKAGE_PIN J1 [get_ports {RA[0]}]
20 set_property PACKAGE_PIN A13 [get_ports {RA[1]}]
21 set_property PACKAGE_PIN N1 [get_ports {RA[2]}]
22 set_property PACKAGE_PIN J4 [get_ports {RA[3]}]
```

3. FPGA 资源占用率



4. SEA 实验板效果图



共有 4 种状态来回切换，约 0.5s 切换一次。

第四部分

总结 /Conclusions

RISC-V 是一种正在风靡全国的流行架构，它对打破 CPU 架构方面的技术壁垒，有着至关重要的作用。而移植 RISC-V 至 SEA board 便是学习 RISC-V 和 SEA 板的一箭双雕的捷径。

移植 MCU 至 SEA 板的过程简单来说是将通用处理器内核 core 的软核针对 SEA xc7s15 做针对型的打包后，通过编写顶层 top.v 的代码和关于硬件的约束文件，实现将 SEA 板转化为 pic16 的 MCU。这一过程既涉及到关于 SEA 开发板硬件的相关知识，也关系到关于 CPU 架构的软件知识以及对 verilog 代码的理解，因此进行 RISC-V 的移植工作对知识的广度和深度都有一定要求，对我们这种初学 SEA board 的新手来说，这是一次非常好的学习的机会。我们既学习了有关 FPGA 的软件编程操作，也学习了相关的硬件知识，同时还学习和了解了 RISC-V 指令集架构的 MPU——pic16。

移植的过程，最需要注意的问题，便是兼容性的问题，在封装 IP 时的兼容性选项以及在顶层调用时模块和 IO 口的绑定都很容易出现不兼容的问题。虽然移植的过程几经波折，但经过努力研究和调试，我们最终实现了 SEA 板上的 RISC-V 架构的 MCU，并利用它跑出了经典的流水灯程序。移植工作的内容虽不多，但依旧会遇到许许多多的问题，这也说明了我们对 FPGA 的认识和了解程度还远远不够，今后也要更加努力地学习有关 FPGA 和 RISC-V 的知识，为制作更加复杂和高级的功能做准备。

参考资料

[1] https://github.com/zzFon/RISC_on_FPGA
[2] CPU 自制入门,【日】水头一寿, 赵谦 译
[3] 手把手教你设计 CPU——RISC-V 处理器, 胡振波
[4] 基于 FPGA 与 RISC-V 的嵌入式系统设计
[5] https://blog.csdn.net/iotthings/article/details/83281101
[6] https://www.bilibili.com/video/BV1Tt4y197qU
[7] https://www.bilibili.com/video/BV1pZ4y1H71C?from=search&seid=8705453685559440585
[8] https://blog.csdn.net/u014470361/article/details/81390365
[9] https://www.bilibili.com/video/BV19T4y1u7bY
[10] https://www.eefocus.com/stonestrong/blog/07-10/75421_9f493.html
[11] https://www.bilibili.com/video/BV1pQ4y1M7Zg?from=search&seid=17791265311565296489
[12] https://www.microchip.com/mplab/compilers*
[13] https://blog.csdn.net/jyunefe/article/details/80644868*
[14] Microchip PIC16C5X EPROM/ROM-Base 8-bit CMOS Microcontroller Series*
[15] Schematic-of-SEA.pdf, version v1.0, designed by weifeng.zeng
[16] Pin-Definition.xlsx
[17] SEA 开发实验指导书.pdf, Disp301 出品